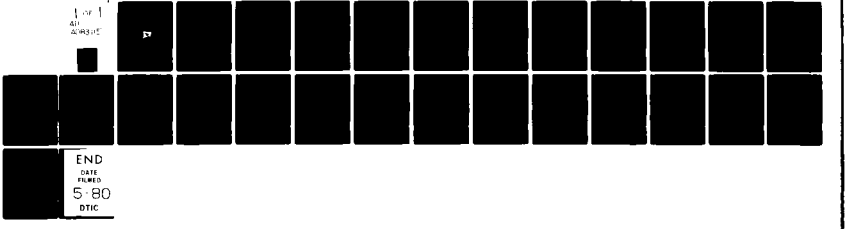
AD-A083 115 BOEING AEROSPACE CO SEATTLE WA BOEING MILITARY AIRPL--ETC F/6 9/2
COMPUTER PROGRAM DEVELOPMENT SPECIFICATION FOR IDAMST OPERATION--ETC(U)
NOV 76 F33615-76-C-1099
UNCLASSIFIED SPEC-SB-4044 AFAL-TR-76-208-ADD-4 NL



END
DATE FILMED
5-80
DTIC

AFAL-TR-76-208, Addendum #4

AD47163
A-559KD

SPECIFICATION
SB 4044

LEVEL III

(6) COMPUTER PROGRAM DEVELOPMENT SPECIFICATION
FOR IDAMST OPERATIONAL TEST PROGRAMS.
Addendum 4.

Prepared by

(18) AFAL
THE BOEING AEROSPACE COMPANY
BOEING MILITARY AIRPLANE DEVELOPMENT
SEATTLE, WASHINGTON
(12) 27

(11) NOV[redacted]76

(19) TR-76-208-ADD-4

(14) SPEC-SB-4044

(15) F33615-76-C-1099

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

PREPARED FOR

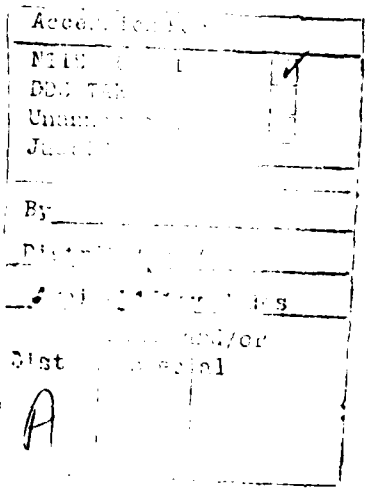
AIR FORCE AVIONICS LABORATORY
AIR FORCE SYSTEM COMMAND
UNITED STATES AIR FORCE
WRIGHT-PATTERSON AFB, OHIO 45433

80 4 15 057

410258

TABLE OF CONTENTS

LIST OF FIGURES



LIST OF TABLES

1.0 SCOPE

1.1 IDENTIFICATION

This specification describes the two computer programs which accomplish the ground readiness testing of the IDAMST system. These two ground test programs will be referred to as GTP-1 and GTP-2.

1.2 FUNCTIONAL SUMMARY

~~The paragraphs below specify~~ the functional performance requirements design constraints and standards to be used in developing the IDAMST ground test computer programs ~~GTP-1 and GTP-2.~~ These computer programs will be executed in the IDAMST core processors prior to a mission in order to accomplish the operational readiness testing of the IDAMST core elements and various IDAMST subsystems. ~~to display the equipment status to the operator, and to initialize the data tables for the in-flight software which describe the equipment status.~~

2.0 APPLICABLE DOCUMENTS

2.1 GOVERNMENT DOCUMENTS

2.1.1 Appendices to Contract F33615-76-C-1099, Statement of Work (SOW)

a. Appendix A - "AMST Mission Profile and Scenario (Updated)".
b. Appendix C - "System Architecture".
c. Appendix E - "DIAS Mission Software OFP Applications (SA-201-303)", 17 June 1976
d. Appendix F - "DAIS Mission Software, Executive (SA-201-320)", 26 December 1975.
e. Appendix H - "Software Management Plan".
f. Appendix M - "TRW System Backup and Recovery Strategy (TRW 6404-5-6-06)", Dept. 1975.

2.1.2 DAIS Documents (Reference)

a. ICD - Mission Operation Sequence: Pilot/Controls and Displays/ Interface with Application Software (SA-803-200), 15 March 1976.
b. Mission Software/Controls and Displays Interface (SA-802-301), 12 March 1976.
c. DAIS System Control Procedure, (SA-100-101 Appendix A), 7 Nov. 75.

2.1.3 IDAMST Documents (Program Generated)

a. Computer Program Development Specification, IDAMST OFP Executive, (SB-4040-41), July 1976.
b. Computer Program Development Specification, IDAMST OFP Error Handling and Recovery (SB 4040-43), July 1976.
c. Computer Program Development Specification, IDAMST OFP Applications (SB-4040-42), July 1976.

2.1.4 IDAMST Documents (Reference)

The following documents because of release dates, serve only as reference documentation for this specification; however, are considered prime to further definition of the IDAMST system design.

a. System Specification for IDAMST, Type A (S1-1010), June 1976.
b. Prime Item Development Specification, IDAMST Processor, Type B1 (S1-4030), June 1976.
c. System Segment Specification, IDAMST Control/Display Subsystem. Type A (S1 5020), June 1976.
d. System Specification, IDAMST Information Transfer System, Type A (SS 3020), May 1976.
e. Prime Item Development Specification, IDAMST Remote Terminal, Type B1 (SS 3130), May 1976.
f. Prime Item Development Specification, IDAMST Bus Control Interface, Type B1 (SS 3230), May 1976.

3.0 REQUIREMENTS

The program GTP-1 shall interface with the IDAMST core elements: The processors, BCIU's, MUX, RT's, displays, MPDG's, DSC, refresh memory, switch matrix, control panels and the mass memory. The function of GTP-1 is to test these core elements, detect any malfunctions, isolate the fault to the lowest LRU level and display the test results on the center MPD. Operator interaction in the form of switch activation and displays presentation verification will be required to perform these tests.

The program GTP-2 shall interface with the IDAMST core elements and with various IDAMST subsystems which are connected with the IDAMST bus. The function of GTP-2 is to test the various IDAMST subsystems, detecting malfunctions, isolating faults to the LRU level and displaying the test results on the center MPD. Operator interaction in the form of switch activation and subsystem observation will be required to perform those tests.

Both GTP-1 and -2 must interface with and be co-resident with the Operational Flight Program (OFP) Executive which is required in order to have interaction with the IDAMST core elemtns and the subsystems. Both tests must also interface with the mass memory, which contairs the tests and on which the test results will be recorded.

No external test equipment will be used in the execution of the test programs GTP-1 and GTP-2.

3.1 COMPUTER PROGRAM DEFINITION

The GTP-1 and GTP-2 programs shall provide the testing of the IDAMST core elements and subsystems, respectively. This testing shall be accomplished using no external equipment. Test results shall be displayed on the center MPD. Operator interaction will be required to control the tests and to indicate certain test results.

3.1.1 Interface Requirements

The GTP programs shall provide software control of the IDAMST core elements and the subsystems that are tested. In addition to interfacing with the core elements and the subsystems, there is a software interface with the Operational Flight Program (OFP) Executive.

3.1.1.1 Interface Block Diagram

Figure 3.1.1-1 illustrates the GTP-1/core element interface. Figure 3.1.1-2 illustrates the GTP-2/subsystem interface. Figure 3.1.1-3 illustrates the GTP/OFP executive interface.

3.1.1.2 Detailed Interface Requirements

This section contains the detailed interface requirements of the GTP programs.

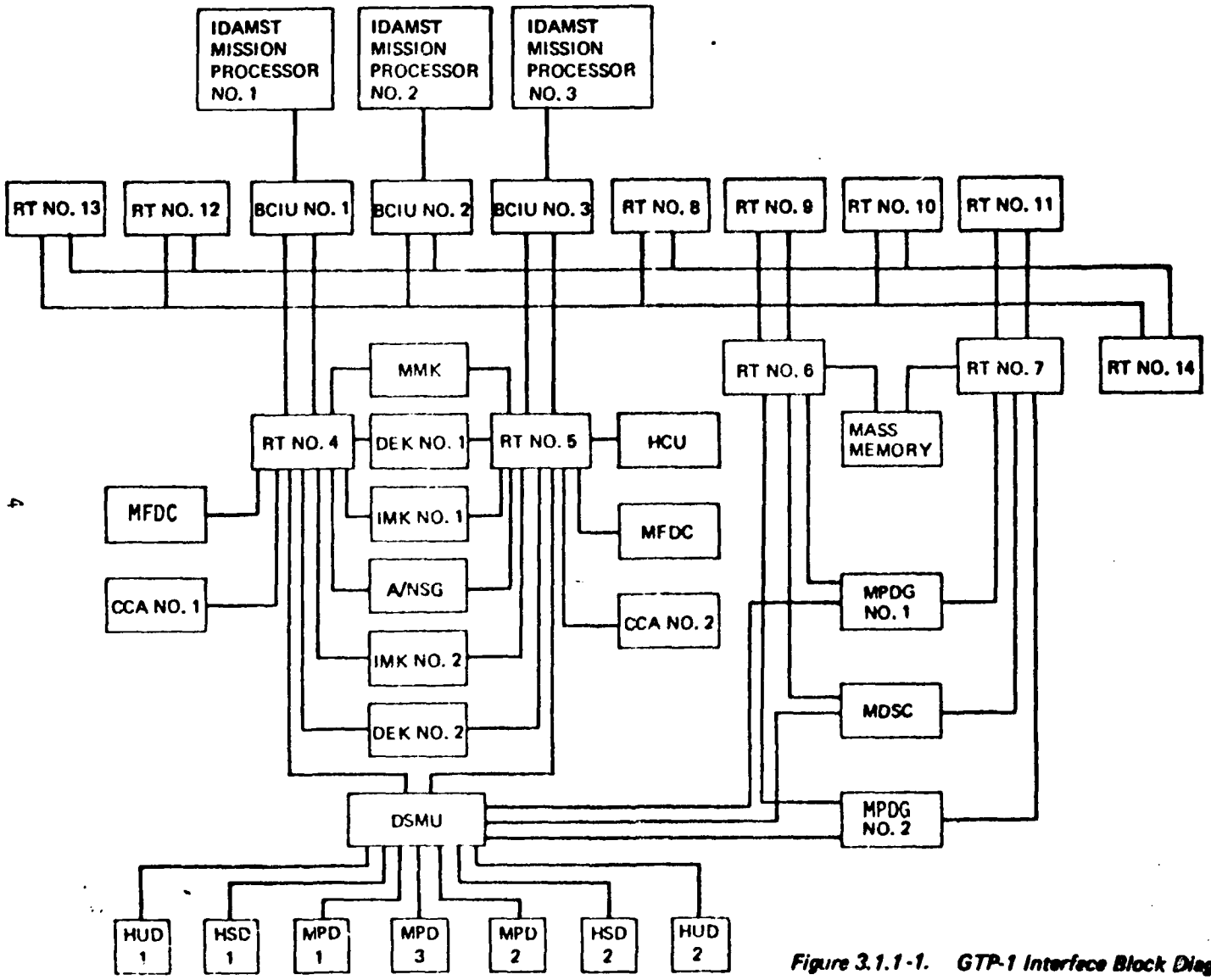


Figure 3.1.1-1. GTP-1 Interface Block Diagram

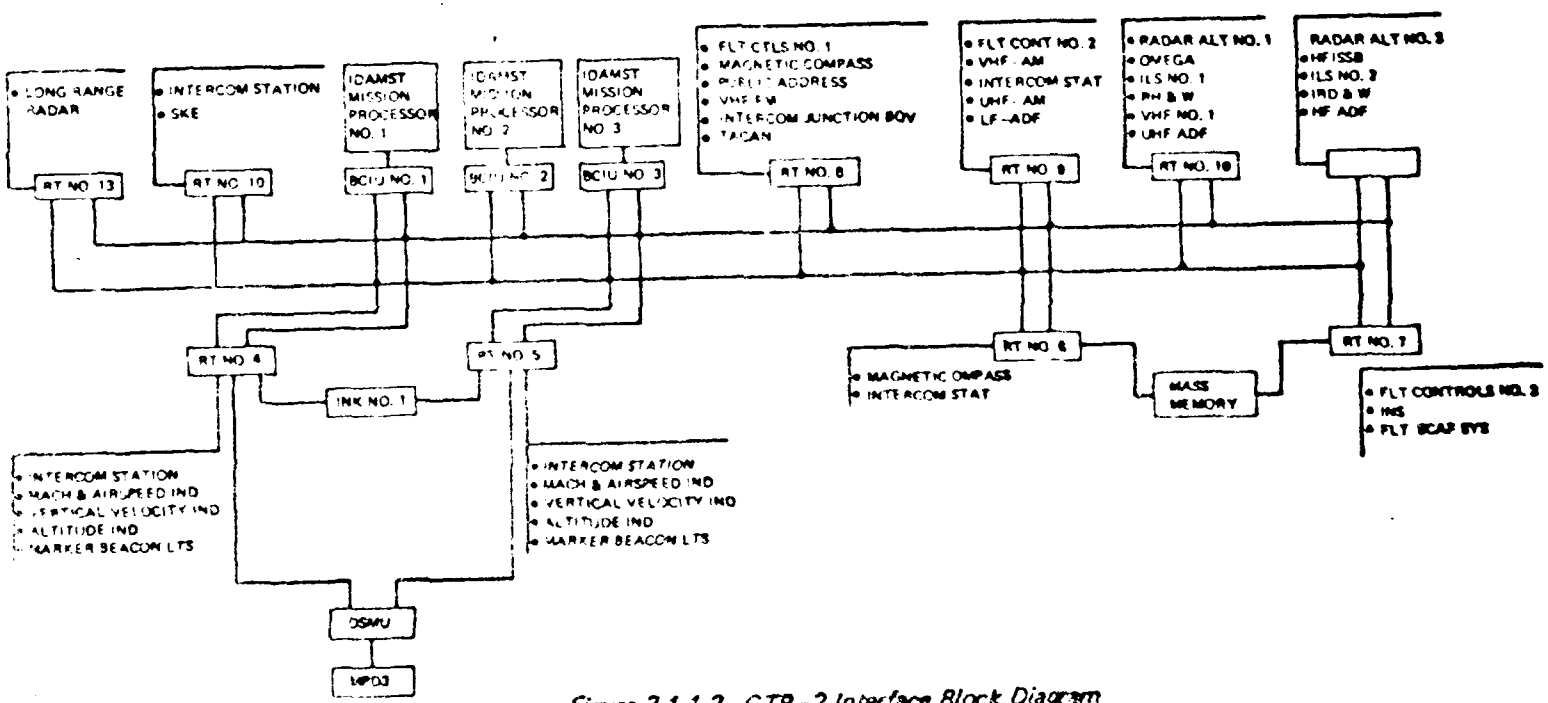


Figure 3.1.1-2. GTP–2 Interface Block Diagram

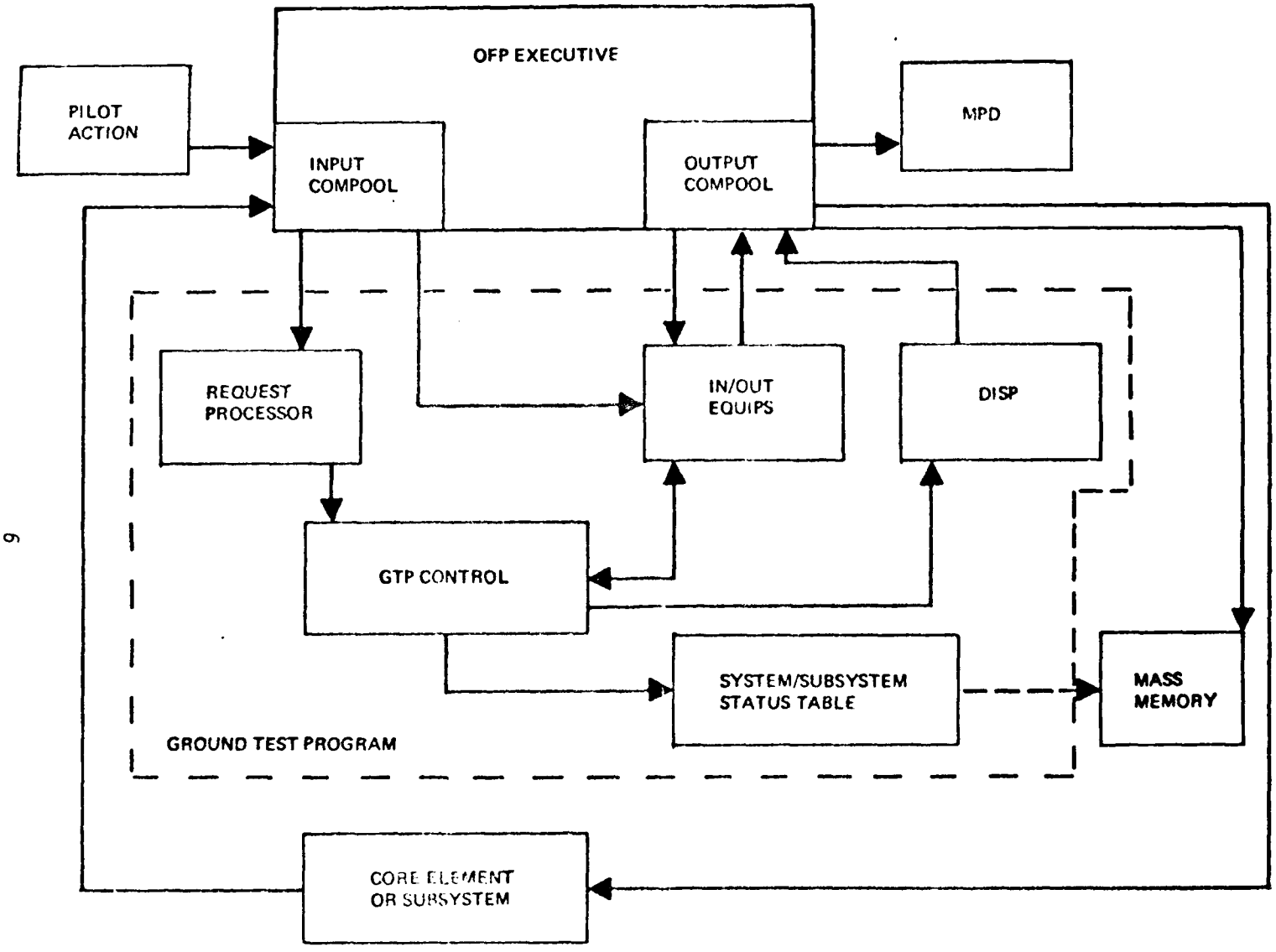


Figure 3.1.1-3. GTP/OFP Executive Interface Block Diagram

3.1.1.2.1 GTP-1

The detailed interface requirements for GTP-1 are contained in IDAMST Operational Flight Program Executive specification.

3.1.1.2.2 GTP-2

The detailed interface requirements for GTP-2 are contained in IDAMST Application Software Specification.

3.1.1.2.3 GTP/Executive Interface

The GTP's interface with the Executive is by three means:

a. Request Processor

b. Input Compool

c. Output Compool

The Request Processor determines which key the pilot pressed and passes this information on to the GTP control program.

The input compool contains all data input by the Executive and is accessed by an input equipment process. This process provides the GTP control program with the needed input data.

The output compool contains all data to be output by the Executive. An output equipment process or a display process transfers data from the GTP control program into this compool.

3.2 DETAILED FUNCTIONAL REQUIREMENTS

3.2.1 GTP-1 Control Program

3.2.1.1 Inputs

The data inputs to the GTP-1 control program shall consist of:

a. Operator execution command (i.e., repeat, skip or continue)

b. Lower level program test results

3.2.1.2 Processing

The GTP-1 control processing shall consist of scheduling the lower level test programs under its control, and the displaying of the test result data. The results of the individual tests shall be displayed on the center MPD, stored in the Subsystem Status Table. At the completion of GTP-1, this Table shall be recorded on the mass memory device.

The test programs, listed in Table 3.2.1.2-1, shall be called in a preplanned order, Table 3.2.1.2-2, and each test shall run to completion. After the completion of each test program, the operator shall have the ability to repeat the test just completed, skip the next scheduled test or continue in the normal sequence.

3.2.1.3 Outputs

The outputs of the GTP-1 control program shall consist of:

a. Test program activation command

b. Test program identification and test results to the MPD

c. Next test program identification to the MPD

d. Subsystem status table to mass memory

3.2.1.4 Special Requirements

The GTP-1 control program shall have a higher priority than the test programs and shall schedule them with timeouts to prevent one of the test programs from being trapped as a result of a faulty LRU.

3.2.2 Master Processing Instruction Test

3.2.2.1 Inputs

The external data input for this test shall be the master processor number.

3.2.2.2 Processing

The Master Processor Test shall exercise each instruction and each control path in the processor and check for errors. Any error shall constitute a failure.

Master Processor Instruction Test

Master Processor Memory Test

Master BCIU, MUX and RT Test

Other BCIU Tests

Other Processor Instruction Tests

Other Processor Memory Tests

Control Panel Tests

Displays, MPDG's, DSC, Refresh Memory and Switch Matrix Tests

Mass Memory Tests

TABLE 3.2.1.2-1
GTP-1 Test Programs

Master Processor Instruction Test

Master Processor Memory Test

Master Processor BCIU, MUX, RT Test

Displays, MPDG's, DSC, Refresh Memory and Switch Matrix Test

Control Panel Tests

Master to Mass Memory Test

Remote (Other) Processor Instruction Test

Remote (Other) Processor Memory Test

Remote Processor BCIU, MUX, RT Test

Remote to Displays, MPDG's, DSC, Refresh Memory and Switch Matrix Tests

Remote to Control Panel Test

Remote to Mass Memory Test

Monitor (Other) Processor Instruction Test

Monitor (Other) Processor Memory Test

Monitor Processor BCIU, MUX, RT Test

Monitor to Displays, MPDG's, DSC, Refresh Memory and Switch Matrix Test

Monitor to Control Panel Tests

Monitor to Mass Memory Test

TABLE 3.2.1.2-2
GTP-1 Test Program Execution Sequence

3.2.2.3 Outputs

The outputs of this program shall be messages which contain the following information:

- o What test has been performed (Processor Test)?
- o Processor number
- o Test result (pass or fail)

3.2.3 Master Processor Memory Test

3.2.3.1 Inputs

The external data input required for this program is the master processor number and the BITE signal from its memory.

3.2.3.2 Processing

The test shall check every memory location in the processor by storing predetermined sequences and reading them, comparing the read value bit for bit with the stored value. For every test sequence chosen, the compliment shall also be used. Any error detected shall constitute a failure.

3.2.3.3 Outputs

The outputs of this test shall be messages containing the following information:

- o What test was performed (memory test)?
- o Processor number tested
- o Test results (pass or fail)

3.2.4 Master BCIU, MUX and RT Tests

3.2.4.1 Inputs

The external data inputs for this test shall be the master processor number and BITE signals from the appropriate IDAMST core elements.

3.2.4.2 Processing

Using this program, the master processor shall test every BCIU instruction and control path in one half of its redundant BCIU. It shall also check every addressable memory location and register in that half of its BCIU writing out and reading in predetermined words. Any error shall constitute a failure in that half of the BCIU. The same tests shall then be repeated for the other half of that BCIU.

The master processor shall next test one half of each RT in turn, using only one of the redundant busses. The BITE from each RT shall be checked first.

The same tests described above shall be used for each RT, checking every RT executable instruction, every control path in that half of the RT, and every addressable memory location and register in that half of the RT. Any error shall constitute a failure of that half of the RT. Response time from each RT shall also be checked and if not within the proper limits, a failure of that half of the RT shall be declared. If none of the RT's on a bus pass the tests, a failure of that bus shall be declared but not the RT's.

The program shall repeat the above tests using the other bus and the other half of each test.

3.2.4.3 Outputs

The outputs of this test shall be messages containing the following information:

- o Type of test reported (BCIU, RT or Bus)
- o Unit number (CIU number, RT number, Bus Number)
- o Pass or fail (or in the case of RT or BCIU, the report can be half fail)

3.2.5 Other BCIU Tests

3.2.5.1 Inputs

The external inputs for this test shall be the master processor number and the BITE signals from the non-master BCIU's.

3.2.5.2 Processing

The master processor shall test one half of each non-master BCIU in turn, using only one of the redundant busses. First, the BITE from each BCIU shall be checked. Then, every BCIU executable instruction, every control path in that half of each BCIU and every memory location and register in that half of each BCIU shall be tested. Any error shall constitute a failure of that half of the BCIU. Response times of each BCIU shall also be checked, and if out of tolerance, a failure of that half of the BCIU shall be declared. If none of the BCIU's on a particular bus respond or pass the tests, a failure of that bus shall be declared.

The test shall then be repeated using the other bus on the other half of each BCIU.

3.2.5.3 Outputs

The outputs shall be messages containing the following information:

- o Type of test (BCIU or Bus)
- o BCIU number tested (or failed bus number)
- o Test results (pass, fail or half fail)

3.2.6 Other Processor Instruction Tests

3.2.6.1 Inputs

The external inputs for this test shall be the master processor number and the BITE signals from the non-master processors.

3.2.6.2 Processing

The GTP-1 Control Program shall test the other (non-master) processors sequentially. First, the BITE signal from the processor will be checked after which the same self test routine that was used to test the master processor, 3.2.2, will be loaded and initiated in the processor to be tested.

3.2.6.3 Outputs

The outputs shall be messages containing the following information:

- o Type of test (processor test)
- o Processor number
- o Test results (pass or fail)

3.2.6.4 Special Requirements

Since there are redundant paths to get the processor test into the processor, the route chosen will be one previously verified to be operational.

3.2.7 Other Processor Memory Tests

3.2.7.1 Inputs

The inputs for these tests shall be the master processor number and the BITE signals from the other (non-master) processor's memories.

3.2.7.2 Processing

The GTP-1 Control Program shall sequentially test the memories of the other (non-master) processors. First, the BITE signal from the memory to be tested shall be checked. Then, the same routine used to test the master processor memory shall be loaded into the appropriate processor and that processor commanded to test its own memory.

3.2.7.3 Outputs

The outputs of the test shall be messages containing the following information:

- o Type of test (memory test)
- o Processor number tested
- o Test results (pass or fail)

3.2.7.4 Special Requirements

Since there are redundant patns to load the memory test into the processor whose memory is to be tested, the route chosen will be one previously determined to be operational.

3.2.8 Control Panel Tests

3.2.8.1 Inputs

The inputs to these tests shall be the control panel switch positions and the BITE signals from the various control panels.

3.2.8.2 Processing

This program shall test all IDAMST control panel switches and lamps. It will require the interaction of an operator in the cockpit.

The program shall first cycle the BITE signal from each of the panels to determine if a bulb is burned out. If a faulty bulb is discovered on any of the lamp drivers, a test subroutine will be run to discover which bulb in which switch is faulty. This information shall be presented to the operator so tne bulb can be replaced. When all bulbs are operational, the operator shall be given a message on an MPD asking his cooperation in pressing buttons. The operator will be sequenced through pressing every switch on the IDAMST control panels by lighting the lamp in the switch to be pressed. When tne switch is pressed, the lamp will go out and the next switch to be pressed will be illuminated.

3.2.8.3 Outputs

Outputs of these tests shall be messages containing the following information:

- o Test performed (control panel tests)
- o Test results by control panel number
  - Panel number
  - Test completed (yes or no)
  - Switches OK (yes or no; if no, list bad switches)
  - Lamps OK (yes or no; if no, list bad lamps)

3.2.8.4 Special Requirements

This test must contain a time out to allow the test to continue in the case of a faulty switch.

3.2.9 Displays, MPDG's, DSC, Refresh Memory and Switch Matrix Tests

3.2.9.1 Inputs

The external inputs required for these tests are the control panel switch

positions and the BITE signals from the elements under test.

3.2.9.2 Processing

The program shall command the self-test mode in the Digital Scan Converter (DSC) and process the result of the self-test to determine the status of the DSC.

The program shall test the BITE signals of the Modular Programmable Display Generators (MPDGs); MPDG #1 shall then be commanded to display a special test pattern (for example, a grid) which covers the screen and can easily be checked for anomalies. The operator will be required to verify that the pattern appears on the proper Multipurpose Display (MPD). Upon operator verification and depressing the proper button, the other MPDG shall furnish the test pattern and the switch illuminated asking for bus verification. The refresh memory shall be tested by repeating the pattern nine more times, using a different refresh memory each time.

The program shall test the BITE signals of all the displays: Three MPD's; two Horizontal Situation Displays (HSD's); the Head Up Display (HUD); and the Integrated Multifunction Keyboard (IMK). For further testing, a TV-type test pattern shall be displayed and the operator asked to verify linearity, resolution and shades of grey. All the displays and the switching matrix shall be tested sequentially by rotating the pattern from display to display. The operator shall press the switch that indicates that both the switch matrix and display are operational if the pattern checks in linearity, resolution, and shades of grey. If the pattern appears but faulty in detail, the operator shall press the switch that indicates a display failure. If the pattern does not appear at all, the operator should not press any switch. Within four seconds, the program shall reroute the test pattern to the display using a different switching matrix setting. If no signal routing succeeds in placing a pattern on the display, that display shall be declared faulty. If one route through the switching matrix consistently fails to produce patterns on any display, the switching matrix shall be declared faulty.

3.2.9.3 Outputs

The outputs of the program shall be messages containing the following information:

- LRU type tested - MPSG, DSC, refresh memory, display or switch matrix
- LRU number (for example, MPD #3)
- Test result (pass or fail or partial fail for refresh memory)

3.2.10 Mass Memory Test

3.2.10.1 Inputs

The inputs to this program shall be the BITE signals from the LRU's in the

mass memory system and the control panel switch positions.

3.2.10.2 Processing

The program shall test the BITE signal from each LRU in the mass memory system. The first word of each data file in the mass memory which is addressable from the master processor shall be read and compared with stored data. If an error is retested, the word shall be reread. Three successive errors shall be sufficient to declare that file record faulty.

The Inflight Recorder shall be tested by recording messages on each available track, reading the messages back and comparing for errors. The same failure criteria as used above shall be used in this test.

3.2.10.3 Outputs

The outputs of this test shall be messages containing the following information:

- o Test name (mass memory)
- o LRU number (example, recorder number)
- o Test results (pass or fail)

3.2.11 GTP-2 Control Program

3.2.11.1 Inputs

The inputs to the GTP-2 control program shall consist of the following:

a. Subprogram execution command (i.e., repeat, skip or continue)

b. Subprogram test results

3.2.11.2 Processing

The GTP-2 control program processing shall consist of sequencing through the GTP-2 subprograms as commanded by the operator and displaying the test results. The results of the individual tests shall be displayed on the center MPD, and stored in the subsystem status table.

The sequencing of the subprograms shall be in a predefined order. After the completion of each subprogram, the operator shall have the ability to repeat the test just completed, skip the next scheduled test or continue in the normal sequence.

3.2.11.3 Outputs

The outputs of the GTP-2 control program shall consist of:

a. Subprogram activation command

b. Subprogram identification and test results to the MPD

c. Next subprogram identification to the MPD.

d. Subsystem status table to mass memory.

3.2.11.4 Special Requirements

The GTP-2 control program shall have a higher priority than the subprograms and shall schedule them with time outs to prevent one of the subprograms from being trapped as a result of a faulty LRU.

3.2.12 GTP-2 Subprograms

This section discusses the subprograms of GTP-2. These subprograms are grouped into three categories according to the type of verification performed:

a. BITE (built-in test equipment) verification

b. Software verification

c. Operator verification

Each item shall be a separate subprogram.

3.2.12.1 BITE Verification Subprograms

The subprograms in this category shall test those subsystems that have BITE information available. These subsystems include:

a. HF/SSB

b. IFF

c. OMEGA Receiver

d. TACAN

e. IMS

f. Weather Radar

g. Radar altimeter

h. Station keeping equipment

i. Flight control system

3.2.12.1.1 Inputs

The inputs to these subprograms shall be the BITE information from the subsystem being tested.

3.2.12.1.2 Processing

The processing of these subprograms shall consist of an evaluation of the BITE information to determine the test results.

3.2.12.1.3 Outputs

The outputs of these subprograms shall be a command to the subsystem to exercise its BITE and the test results to the GTP-2 control program.

3.2.12.2 Software Verification Subprograms

The subprograms in this category shall test those subsystems that supply data to the IDAMST processor. These subsystems include:

a. INS
b. Radar Altimeter
c. Magnetic Compass
d. Flight Control Subsystem

3.2.12.2.1 Inputs

The inputs to these subprograms shall consist of:

a. Data values expected from the subsystem being tested
b. Data values from the subsystem being tested

3.2.12.2.2 Processing

The processing of those subprograms shall consist of comparing the expected data values with the subsystem supplied data values. Errors exceeding a predefined value shall constitute a subsystem failure.

3.2.12.2.3 Outputs

The outputs of these subprograms shall be:

a. A command to the subsystem to supply its data
b. The test results to the GTP-2 control program

3.2.12.3 Operator Verification Subprograms

The subprograms in this category shall test those subsystems which require the operator to determine the correctness of the subsystem response. These subsystems include:

a. UHF-AM
b. VHF-AM
c. VHF-FM
d. HF/SSB
e. Intercommunication Set
f. Public Address Subsystem

g. OMEGA Receiver

h. UHF-ADF

i. LF-ADF

j. TACAN

k. ILS

l. Dedicated Instruments

3.2.12.3.1 Inputs

The input to these subprograms shall be the operator action indicating the test results.

3.2.12.3.2 Processing

The processing of these subprograms shall consist of interpreting the operator input in order to determine the test results.

3.2.12.3.3 Outputs

The outputs of these subprograms shall consist of:

a. A command to the subsystem being tested

b. The test results to the GTP-2 control program

3.3 ADAPTATION

3.3.1 General Environment

The GTP test programs must be consistent with and interface with the OFP. The OFP executive, bus controller, keyboard and display service routines will be needed in performing GTP-1. In addition to the OFP requirements of GTP-1, GTP-2 will require the executive sensor management.

GTP program loading will be initiated at the processor control panel with execution starting automatically after a successful load.

3.3.2 System Parameters

The parameters and constants associated with each test shall be stored in the software module containing that test. The modular software construction ensures that if equipment is added or changed, or new modes defined, the necessary parameters can be changed by changing only those software modules directly associated with the equipment changes.

3.3.3 System Capacities

The total number of test modules is not specified so that tests can be added or deleted as the system evolves. Provision is made in both GTP-1 and GTP-2 for additional tests that are not normally a part of the ground test sequence.

The number of parameters, data rates, storage capacities, etc., are not specified and are limited only by interfacing equipment capacities.

4.0 QUALITY ASSURANCE PROVISIONS

This section identifies the basic method for accomplishing software verification.

4.1 Introduction

IDAMST CPCIs will incorporate top-down, structured concepts, described briefly below:

Structured Program

A structured program is a computer program constructed of a basic set of control logic figures which provide at least the following: Sequence of two or more operations, conditional branch to one of two operations and return repetition of an operation. A structured program has only one entry and one exit point. A path will exist from the entry to each node and from each node to the exit. In addition, certain practices are associated, such as indentation of source code to represent logic levels, use of intelligent data names and descriptive commentary.

Top-Down Programming

Top-down programming is the concept of performing in hierarchical sequence a detailed design, code, integration and test as concurrent operations.

Top-Down Structured Programs

A top-down structured program is a structured program with the additional characteristics of the source code being logically but not physically segmented in a hierarchical manner and only dependent on code already written. Control of execution between segments is restricted to transfers between vertically adjacent hierarchical segments.

Top-down coding and verification is an ordering of system development which allows for continual integration of the system parts as they are developed and provides for interfaces prior to the parts being developed. At each stage, the code already tested drives the new code, and only external data is required.

In top-down programming, the system is organized into a tree structure of segments. The top segments contain the highest level of control logic and decisions within the program, and either passes control to the next level segments or identifies the next level segments for in-line inclusions. The next level may include stubs. Stubs which are to be replaced eventually with running code may contain a "no operation" instruction or possibly a display statement to the effect that control has been received. The process at replacement of successively lower level stubs with operational code continues until all functions within a system are coded and verified.

In top-down coding and verification, the highest level element is coded first. Coding, checkout, and integration proceed down the hierarchy until the lowest levels have been integrated. This does not imply that all elements at a given level are developed in parallel. Some branches will intentionally be

developed early, e.g., to permit early training and early development of critical functions or hardware/software integration.

Many systems interfaces occur through the data base defintion in addition to calling sequence parameters. Top-down programming requires that sufficient data definition statements be coded and that data records be generated before exercising any segment which references them. Ideally, this leads to a single set of definitions serving all the programs in a given application.

This approach provides the ability to evolve the product in a manner that maintains the characteristic of always being operable, extremely modular and always available for successive levels of testing that accompany the corresponding levels of implementation. Exception to the top-down coding and integration approach will be considered on a case-by-case basis.

Each computer program will be coded in a higher order language. Use of assembly or machine language will be restricted to coding of certain executive functions where the higher order language cannot be used.

Real Time Structured Programs

An additional complexity in the IDAMST system is the Real Time, asynchronous communication of structured programs as tasks. Tasks are also organized as a hierarchy. Each task has a Controller Task which is the only task permitted to schedule or cancel the lower level task. However, any task is permitted to activate any other task in IDAMST.

4.2 Computer Program Verification

Computer program verification is the process of determining whether the results of executing a computer program in a test environment agree with the specification requirements. Verification is usually only concerned with the logical correctness of the computer program (i.e., satisfying the functional/performance requirements) and may be a manual or a computer-based process (i.e., testing software by executing it on a computer).

The use of top-down structured programming techniques provide certain program characteristics that may lead to a simplification of the computer program verification process. Top-down integration of the program elements in a CPCI minimizes the use of complex driver routines and replaces them with actual program elements and simple program stubs. It also provides a system in which the computer program is continually being tested as successively lower levels of program elements are integrated and the interfaces between program elements are verified prior to the integration of the next lower level.

4.2.1 Program Element Tests

Program elements are coded in the sequence required for top-down integration. When coding and code review are completed, each program element shall be functionally tested in a stand-alone configuration by the programmer to assure that the element can be executed and that the specified functions are performed. Since program elements are small and are restricted to one entry point and one exit point, the test environment is relatively simple.

4.2.2 CPCI Integration Tests

Following successful completion of the Program Element Tests, the program elements are entered into the Computer Program Library where they are subjected to configuration control procedures. Controlled program elements are compiled/assembled, link-edited and the current CPCI version is made available for integration testing. Integration tests are dynamic tests designed to verify program functions and interfaces between program elements and with the data base. The result is a complete CPCI for which all design features have been verified.

The integration of program elements or tasks into the complete computer program shall be accomplished in a top-down sequence. The highest level elements which contain the highest level controller tasks shall be tested and integrated first. These tasks are the Master Sequencer, Configurator, Request Processor, and Subsystem Status Monitor. Testing and integration shall proceed down the hierarchy until all program elements (e.g., equipment interface functions), have been integrated and the design completely verified.

An important aspect of integration testing of IDAMST will be the invocation and synchronization of the tasks, since these functions do not fall under the structured programming rules.

4.2.3 Formal Software Testing

The purpose of formal testing is to confirm that the computer program performs the functions and satisfies the performance requirement contained in the software requirements specification. Formal testing consists of Preliminary Qualification Tests (PQT) and Formal Qualification Tests (FWT), and are conducted in accordance with Air Force approved test plans.

Pre-Qualification Testing (PQT)

PQT is an incremental process which provides visibility and control of the CPC2 development during the time period between the Critical Design Review and Formal Qualification Testing.

PQT consists of functional level tests, conducted at the development facility, and using Air Force approved test plans. These tests will use documented procedures, completed by the contractor, and submitted to the Air Force Sufficiently in advance of the scheduled test session to permit review and analysis. They will typically use controlled inputs specifically prepared for the test purpose.

A Pre-Qualification test will generally be conducted for each CPCI function. If a test's cost or time consumption estimates are significantly high, the test will be deferred to FQT unless it is time-critical or performance-critical to the development of the CICI.